1Г

УДК 519.71

С.В. Гафуров, В.В. Краснопрошин
Белорусский государственный университет, г. Минск, Республика Беларусь
gafurov@bsu.by, krasnoproshin@bsu.by

Программная технология построения систем для решения задач распознавания со сложной структурой

В работе предлагается программная технология разработки прикладных систем для решения задач распознавания определенного класса. Описан класс задач, допускающих неполноту информации и сложную структуру знаний в предметной области. Предложены алгоритмы, состав и архитектура системы.

Введение

Существуют практические задачи, которые могут быть сформулированы в терминах задачи распознавания образов (ЗРО). Это различные задачи медицинской и технической диагностики, задачи поддержки принятия решений в управлении и т.д.

Несмотря на существование областей практического применения, достаточно развитой математический аппарат и длинную историю развития, количество прикладных систем, основанных на использовании моделей и методов ЗРО сравнительно невелико. Одной из причин такого положения является несоответствие между уровнем затрат на разработку и эффектом, получаемым от использования такой прикладной системы.

Однако современные коммуникационные технологии позволяют существенно увеличить количество пользователей программной системы, устраняя территориальные ограничения. Сегодня можно создавать системы, доступные пользователям по всему миру. Совершенствуются и инструментальные средства разработки. При таких условиях возрастает количество практических задач, для решения которых оправдана разработка автоматизированной системы. В свою очередь, большое количество задач делает актуальной разработку программной технологии, которая бы позволяла строить однотипные системы при небольших затратах.

В работе рассматривается класс задач и алгоритмическая база для их решения в терминах ЗРО. На основе рассмотренных алгоритмов предлагается архитектура и состав программной технологии, которая позволяет достаточно просто (в короткий срок и с низкими затратами) строить прикладные системы. К разработке привлекается специалист предметной области, а труд программиста практически исключен.

Класс задач

Рассмотрим практические задачи, которые могут быть сформулированы в терминах задачи распознавания образов (ЗРО).

В стандартной постановке ЗРО рассматривается множество объектов D, в котором существует L классов $K_1, K_2, ..., K_L$. Произвольный объект множества D описывается вектором x из R^n. Его координаты иногда называют признаками.

Информация о классах задается в виде обучающей выборки или с помощью правил. Требуется построить алгоритм А, который для входного объекта x (на основании обучающей выборки) получает некоторый вектор y, значение которого можно интерпретировать в терминах принадлежности x к классам $K_1, K_2, ..., K_L$.

На практике за такой, достаточно общей, постановкой может скрываться задача медицинской диагностики, где классы – это диагнозы, а в векторе x кодируется результат обследования пациента – симптомы, результаты тестов и анализов. Возможно и бытовое применение. Например, советующая вопросно-ответная система на Интернет-сайте, которая помогает в линейке продуктов выбрать один или несколько товаров, наиболее точно соответствующих потребностям клиента.

При решении практических задач приходится учитывать особенности предметной области, которые не отражены в классической постановке. Одной из таких особенностей является неполнота входной информации [1], которая выражается в отсутствии значений некоторых признаков в векторе x. Другая характерная черта многих практических задач – сложность предметной области, наличие связей между классами и связей между признаками, которые не могут быть отражены только лишь одной обучающей выборкой [2]. Зачастую присутствуют определенные ограничения, связанные с отсутствием ресурсов на сбор информации по всему вектору x.

В данной работе ограничимся рассмотрением задач, для которых характерны именно эти особенности – неполнота входных данных и/или сложная структура информации в предметной области. Оба ограничения требуют разработки специализированных алгоритмов и схем для решения задачи.

Требования к программной технологии

Будем рассматривать программную технологию как средство построения автоматизированных систем, предназначенных для решения практических задач из описанного выше класса. В основу концепции технологии положены следующие идеи.

Во-первых, использование единого для всех решаемых задач алгоритмического ядра, построенного на основе параметрического семейства алгоритмов.

Во-вторых, наличие в технологии инструментальных средств, которые позволяют специалисту в предметной области (эксперту) самостоятельно и достаточно просто строить прикладную систему, не прибегая к помощи программиста.

Под технологией будем понимать набор программных инструментов, модулей и методов их применения для построения прикладной системы. Результатом однократного применения технологии является прикладная автоматизированная система.

Такая система должна:

– решать задачу из описанного класса, т.е. осуществлять сбор информации об объекте x в терминах рассматриваемой предметной области и относить x к одному из классов $K_1, K_2, ..., K_L$, определенных для решаемой задачи;

– обеспечивать информационную поддержку всех этапов решения задачи, т.е. снабжать пользователя информацией, поясняющей текущий этап сбора информации или полученные системой решения;

– быть простой в использовании, т.е. обладать доступным интерфейсом, пригодным для непрофессионального пользователя;
– работать в распределенной среде, т.е. в сетях типа intranet и в Internet;
– обеспечивать одновременную работу нескольких пользователей с системой.

Многие из приведенных выше требований ориентируют разрабатываемую прикладную систему на массовое использование.

Сама технология должна:
– содержать программные модули, составляющие ядро описанной выше системы;
– инструменты для настройки системы на предметную область, т.е. имеются в виду как инструменты для описания классов, признакового пространства и обучения в смысле ЗРО, так и информационное наполнение системы;
– инструменты для сборки готовой автоматизированной системы из названных модулей и информации о предметной области, инструменты для отладки.

Алгоритмическое ядро системы

Основой описываемой системы являются алгоритмы решения ЗРО из заданного класса. Выделим некоторые особенности практических задач, которые не укладываются в классическую постановку ЗРО. Первая из них – неполнота входной информации, вторая – сложная, в некотором смысле, структура знаний в предметной области.

Рассмотрим проблему неполноты входной информации. Пусть D – множество допустимых объектов $x \in R^n$. Под неполнотой понимается отсутствие информации о значении некоторых признаков, т.е. пропуски в векторе x, описывающем объект предметной области. Факт отсутствия информации можно кодировать некоторым специальным значением, например, 0.

В общем случае невозможно искать точное решение на неточных данных. Для решения задачи в этих условиях предлагается применять алгоритм, описанный в [1]. Алгоритм A построен в виде $A = R \circ r$, где R и r – соответственно распознающий оператор и решающее правило [3], [4]. Распознающий оператор R для допустимого объекта x и класса K_j вычисляет интервал $[R^{j-}(x), R^{j+}(x)] \subseteq [-1, 1]$, $j = \overline{1, L}$. Границы интервала определяют максимальную и минимальную достоверность принадлежности объекта классу. В [1] показано, что при последовательном устранении неопределенных значений (сборе информации) границы этого интервала монотонно не расширяются.

Первое свойство алгоритма – способность давать оценку даже при минимальной информации. А монотонное изменение оценок при поступлении дополнительной информации позволяет строить монотонные, в некотором смысле решающие, правила – при поступлении дополнительной информации множество классов решений не расширяется (сужается).

На практике свойство монотонности дает возможность не вычислять точного значения распознающего оператора, а ограничиться некоторым его интервалом. Это может оказаться существенным, к примеру, когда цена получения информации об объекте является высокой, или достижение точного результата требует чрезмерных ресурсов.

Во многих задачах знания, описывающие предметную область, имеют сложную структуру. Это выражается:
– в наличии дополнительных связей между классами;
– в объединении признаков в группы, внутри которых семантические связи сильнее, чем связи между признаками из разных групп.

Таким предметным областям также характерна большая размерность признакового пространства. Обследование, т.е. сбор информации по признакам, становится ресурсоемкой задачей. Крайне желательно в таком случае направлять сбор информации и получать оптимальное в некотором смысле решение до наступления полного обследования.

Предлагается подход, рассмотренный в работе [2], основанный на сведении решения исходной ЗРО Z большой размерности к последовательному решению ЗРО меньшей размерности. Постулируется, что в предметных областях со сложной структурой существует разбиение исходной задачи на подзадачи. Задачи вместе со своими локальными классами образуют граф, определяющий частичный порядок на информации.

На рис. 1 схематично показана идея такого разбиения. Исходная задача Z заменена на задачи z_1, z_2, z_3 и z_4 меньшей размерности. Множество признаков P_i задачи z_i является подмножеством множества признаков P задачи Z. Вершины p_i и f_j – классы в своих задачах. Существует соответствие между классами f_j задач z_i и классами $K_1, K_2, ..., K_L$.

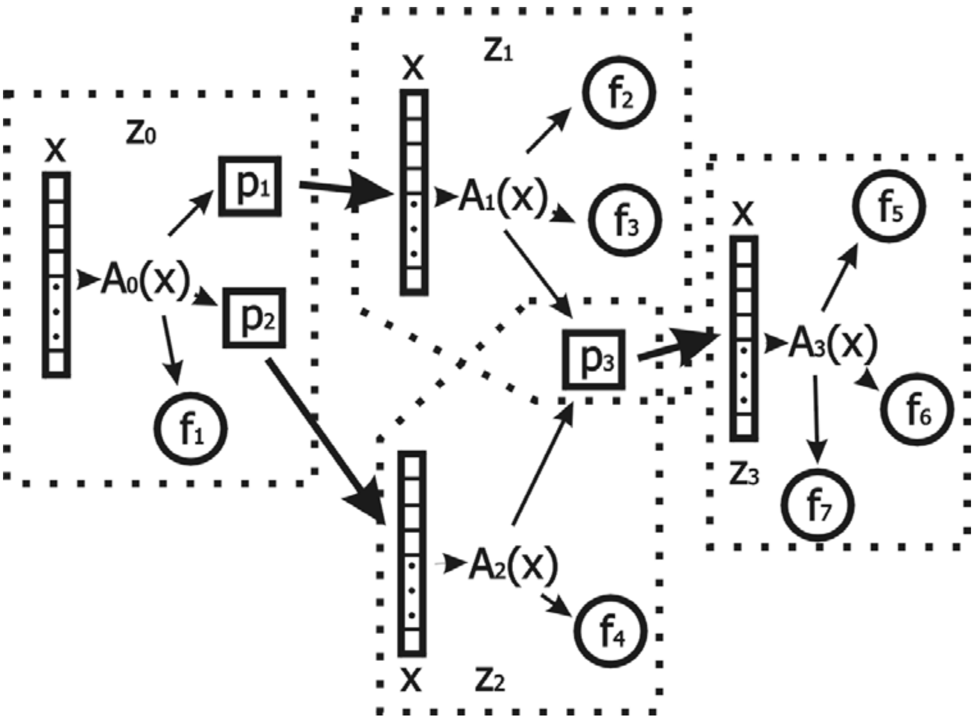


Рисунок 1 – Разбиение задачи на подзадачи

В [2] предлагается формальная процедура, которая:
– определяет очередность решения подзадач, тем самым направляя процесс сбора информации;

– позволяет оценить перспективность построенного на текущий момент решения по отношению к другим возможным решениям;
– обоснованно прекратить поиск решения до момента, когда выполнено полное обследование, получив при этом наиболее достоверное решение.

Архитектура и состав

Для рассмотренного выше класса задач будем строить программную технологию, состоящую из набора программ/инструментов, которые позволяют быстро создать автоматизированную систему для решения той или иной прикладной задачи.

В состав технологии входят компоненты времени выполнения и инструменты этапа разработки. На рис. 2 показан состав функционирующей прикладной системы и основные взаимодействия между модулями системы.

Компоненты времени выполнения:
– решающий модуль;
– интерфейсный модуль;
– модуль семантически привязанной информации;
– файл знаний.

Кроме этого в состав среды времени выполнения включены компоненты «браузер» и «веб-сервер». Они являются стандартными сторонними модулями, использование которых является необходимым (но не достаточным) условием функционирования системы с сети Internet.

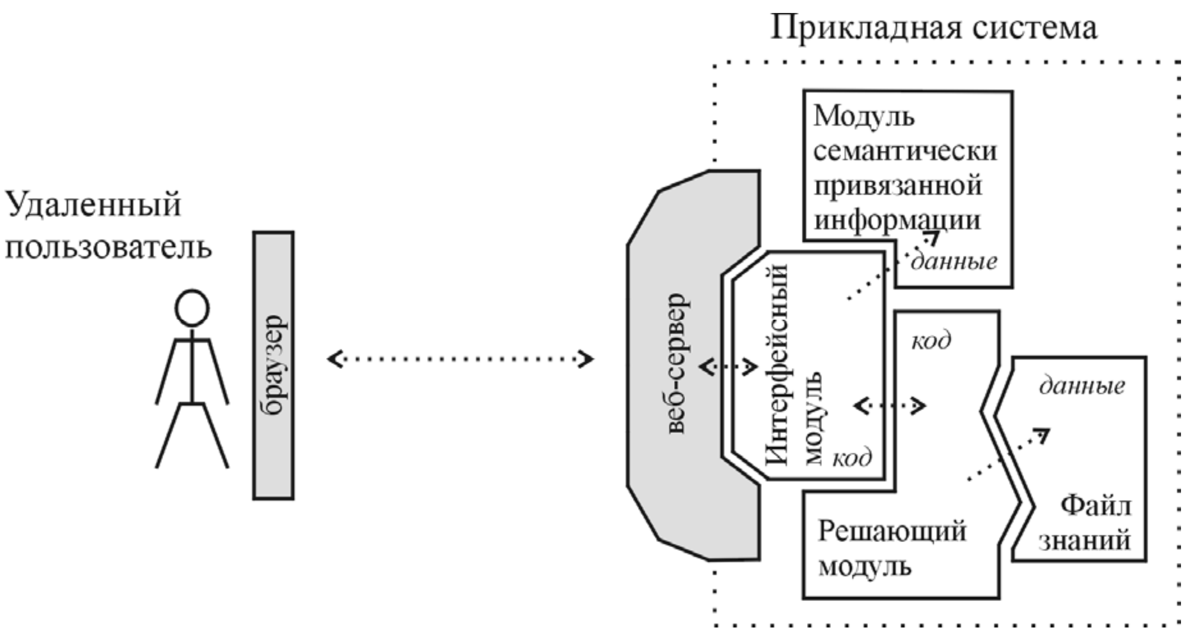


Рисунок 2 – Прикладная система в среде времени выполнения

Сценарий одного цикла работы такой системы следующий. Выполняется обследование – система собирает значения признаков, задавая вопросы пользователю. Получая ответы, система строит решение исходя из собранных на текущий момент данных. Процесс сбора данных продолжается до тех пор, пока не окажется, что получение новых данных не приведет к улучшению полученного решения. В крайнем случае система может произвести полное обследование. Процесс поиска решения

может быть прерван пользователем. В процессе диалога пользователь может запрашивать у системы информацию, контекстно-привязанную к текущему этапу обследования. Эта информация носит, как правило, справочный характер.

Решающий модуль (РМ) – центральная часть прикладной системы, которая реализует алгоритмическое ядро. Он представляет собой исполняемый модуль, используемый в неизменном виде во всех прикладных системах, построенных с использованием рассматриваемой технологии. В ходе решения задачи РМ направляет обследование и отвечает за вычисление оценок. Необходимую для работы информацию он получает из файла знаний.

Интерфейсный модуль (ИМ) обеспечивает диалог с пользователем и выполняет роль посредника между пользователем и РМ. ИМ преобразует требования РМ, выраженные в терминах признакового пространства, в вопросы, понятные пользователю и наоборот – ответы пользователя преобразует в значения признаков. ИМ тоже является исполняемым модулем, общим для всех прикладных систем. ИМ содержит настраиваемую часть, которая позволяет изменять внешний облик системы.

Модуль семантически привязанной информации (МСПИ) хранит статическую информацию справочного характера, соответствующую различным этапам обследования. В отличие от двух предшествующих модулей – это модуль данных. Его содержимое меняется от одной прикладной системы к другой и позволяет настраивать систему на предметную область.

Однако решающую роль в настройке системы на предметную область играет так называемый файл знаний. В этом массиве информации в формализованном виде описано признаковое пространство, перечислены классы, в виде правил и обучающих выборок задано описание классов, указаны дополнительные параметры работы алгоритмов распознавания. Здесь же находится содержательное описание признаков в терминах предметной области, необходимое для ведения диалога с пользователем. Файл знаний активно используется всеми остальными модулями системы.

Модуль семантически привязанной информации отличается от файла знаний тем, что содержит слабоформализованную информацию, предназначенную для обработки человеком, и никак не используется в алгоритмах распознавания. Роль системы в этом случае сводится к предоставлению этой информации пользователю в нужные моменты времени.

Для разработки прикладной системы описанного выше типа необходимы специалист предметной области (эксперт) и веб-дизайнер. В простых случаях достаточно только эксперта. В своей работе эти специалисты используют инструменты времени разработки – это Дизайнер, Отладчик и Компоновщик (рис. 3).

Эксперт создает формальное описание предметной области в файле знаний, а именно: определяет перечень классов, признаковое пространство, вводит обучающую выборку и правила, устанавливает параметры работы алгоритмов распознавания. Для ввода данных он использует «Дизайнер». С использованием другого инструмента, Отладчика, эксперт может проверить работу решающего модуля на введенных в файл знаний данных. Это позволяет укоротить цикл разработки и проверять качество работы на ранних этапах, не выполняя полного развертывания всей прикладной системы.

Веб-дизайнер выполняет две функции. Во-первых, он наполняет модуль семантически привязанной информации данными. В названном модуле хранится текстовая и графическая информация в стандартных форматах (html, jpeg, gif, png и т.п.). Поэтому для ввода информации в МСПИ можно применять сторонние инструменты, работающие с информацией в этих форматах. Во-вторых, веб-дизайнер формирует образ прикладной системы, который впоследствии может быть установлен на веб-сервер. Эту функцию ему помогает выполнять Компоновщик.

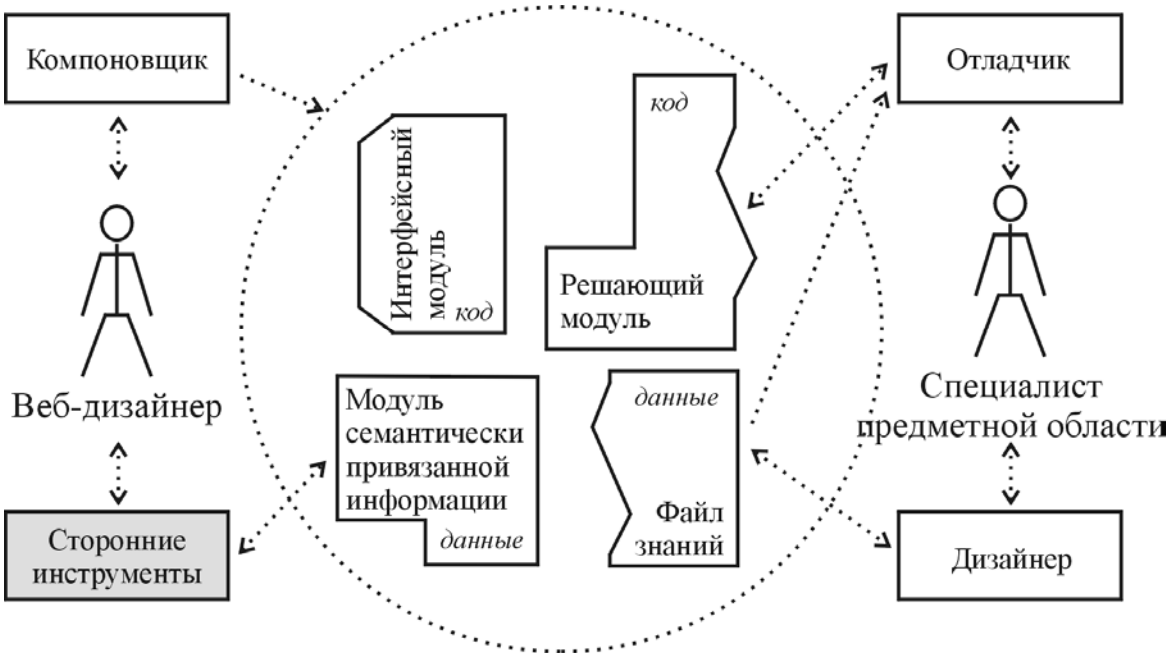


Рисунок 3 – Среда времени разработки

Описываемая технология была реализована, и с ее использованием создан ряд прикладных систем. Например, разработана медицинская система синдромальной диагностики. Названная система ориентирована на неподготовленного пользователя, не связанного с медициной. Путем направленного опроса система пытается обнаружить опасные комбинации симптомов. По итогам обследования система пытается определить заболевание/класс заболевания и может порекомендовать срочно (или не срочно) обратиться к соответствующему специалисту для дальнейшего обследования[1]. Стоит заметить, что приведенный пример интересен с практической точки зрения, но лишь частично использует возможности, заложенные в используемые в системе алгоритмы распознавания.

Заключение

В работе рассмотрена проблема разработки программной технологии, предназначенной для создания прикладных систем, решающих задачи распознавания образов определенного класса. Описан класс задач, в котором допускаются неполнота входной информации и сложная, в определенном смысле, структура знаний в предметной области. Для этого класса предложены соответствующие

[1] Система доступна по адресу: http://www.whatistheproblem.com.

алгоритмы распознавания. Сформулированы требования и описаны архитектура и состав предлагаемой программной технологии. Особенной чертой этой технологии являются низкие затраты на разработку отдельной прикладной системы, что достигается за счет использования унифицированного алгоритмического ядра и специальных инструментов разработки. В частности, наличие последних позволяет специалисту предметной области самостоятельно строить формальное описание предметной области и создавать информационное наполнение системы, не прибегая к помощи профессиональных программистов.

Литература

1. Гафуров С.В., Краснопрошин В.В., Образцов В.А. Проблема неполноты информации в задаче распознавания образов // Вестник БГУ. – Сер. 1. – 2007. – № 2. – С. 113-115.
2. Gafurov S., Krasnoproshin V., Obraztsov V., Vissia H. Pattern Recognition Problem And Partially Ordered Information // Proc. IX International Conf. PRIP’2007. – Minsk. – 2007. – Vol. 2. – P. 131-135.
3. Krasnoproshin V.V., Obraztsov V.A. Problem of Solvalibility and Choice of Algorithms for Decision Making by Precedence // Pattern Recognition and Image Analysis. – 2006. – Vol. 16, № 2. – P. 155-169.
4. Журавлев Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. – М.: Наука, 1978. – Вып. 33 – С. 5-68.

С.В. Гафуров, В.В. Краснопрошин

Програмна технологія побудови систем для рішення задач розпізнавання зі складною структурою

У роботі пропонується програмна технологія розробки прикладних систем для рішення задач розпізнавання певного классу. Описаний клас задач, які допускають неповноту інформації та складну структуру знань у предметній сфері. Запропоновані алгоритми, склад та архітектура системи.

S.V. Gafurov, V.V. Krasnoproshin

Software Technology for Solving Recognition Problems with Complex Structure

The paper considers a software technology which is designed for mass production of applied systems solving pattern recognition problems of a certain class. Authors describe the problem class, which permit incomplete input information and complex knowledge structure of the application domain. Algorithms are suggested. Contents and architecture of the system are described.

Статья поступила в редакцию 13.12.2007.